From Technologies to Solutions

Pluggable Authentication Modules

The Definitive Guide to PAM for Linux SysAdmins and C Developers

A comprehensive and practical guide to PAM for Linux: how modules work and how to implement them

Kenneth Geisshirt

PACKT PUBLISHING

Pluggable Authentication Modules

The Definitive Guide to PAM for Linux SysAdmins and C Developers

A comprehensive and practical guide to PAM
for Linux: how modules work and how to
implement them

Kenneth Geisshirt

PACKT
PUBLISHING

BIRMINGHAM - MUMBAI

Pluggable Authentication Modules

The Definitive Guide to PAM for Linux SysAdmins and C Developers

Copyright © 2007 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, Packt Publishing, nor its dealers or distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: January 2007

Production Reference: 1211206

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-904811-32-9

`www.packtpub.com`

Cover Image by `www.visionwt.com`

Credits

Author

Kenneth Geisshirt

Reviewers

Ralf Hildebrandt
Huang Zhen

Development Editor

Louay Fatoohi

Assistant Development Editor

Nikhil Bangera

Technical Editor

Mithil Kulkarni

Editorial Manager

Dipali Chittar

Project Manager

Patricia Weir

Indexer

Bhushan Pangaonkar

Proofreaders

Martin Brooks
Chris Smith

Layouts and Illustrations

Shantanu Zagade
Manjiri Nadkarni

Cover Designer

Shantanu Zagade

About the Author

Kenneth Geisshirt is a chemist by education, and is a strong free software advocate. He spent his Christmas holidays in 1992 installing SLS Linux, and GNU/Linux has been his favorite operating systems ever since.

Currently, he does consultancy work in areas like scientific computing and Linux clusters. He lives in Copenhagen, Denmark with his partner and their two children. You can find him at `http://kenneth.geisshirt.dk/`.

About the Reviewers

Ralf Hildebrandt is an active and well-known figure in the Postfix community, working as a systems engineer for T-Systems, a German telecommunications company.

He speaks about Postfix at industry conferences and hacker conventions and contributes regularly to a number of open source mailing lists. Ralf Hildebrandt is co-author of *The Book of Postfix*.

Huang Zhen is a software engineer at IBM China Development Labs.

He has been working on the Linux-HA project since 2004 and contributed several components to the project.

PAM-related functions in the Linux-HA project were developed by him.

Table of Contents

Preface 1

Chapter 1: Introduction to PAM 7

- History of PAM 7
- PAM Solves the Authentication Problem 8
- Need for PAM 9
- Installing Linux-PAM 9
  - Downloading 10
  - Compiling 10
  - Extra Modules 12
- PAM Implementations 12
- Summary 14

Chapter 2: Theory of Operation 15

- PAM File System Layout 15
- The PAM Framework 17
- Online Documentation 19
- Services 20
- Management Groups 22
  - The Auth Group 23
  - The Account Group 24
  - The Session Group 25
  - The Password Group 25
- Stacking 26
- Control Flags 26
  - Requisite 27
  - Required 27
  - Sufficient 27
  - Optional 28
  - Order matters 28

Consolidating Your PAM Configuration **29**
Securing Your Environment **32**
An Example 32
Summary **35**

Chapter 3: Testing and Debugging

When the new module is compiling without errors, it must be tested. A simple testing method is to set up a test service and use the pamtester utility as described in Chapter 3.

Summary

Developing with PAM might be a new world for you. But if you are used to UNIX programming in C, it is not a completely strange world.

Applications can easily become PAM-aware, and that might give your applications a great deal of flexibility when it comes to authentication. Moreover, in situations where you cannot find a PAM module that satisfies your demands, it is possible to write your own module.

A

Source code

The C programming language is a natural choice when developing either PAM-aware applications or new modules. Chapter 6 shows examples of an application and a module, and in this appendix you find the source code for both examples.

Vault – Secure Database

The vault program is an example of a PAM-aware application. It provides access to a small database where users can store key/value pairs. The database behind vault is the GNU dbm, and it is not a sophisticated usage of it. The program is somewhat dependent on Linux-PAM due to the fact that the program uses the conversation function provided by Linux-PAM.

```
/*
 * vault.c - access to a secure data vault
 *
 * Kenneth Geisshirt <http://kenneth.geisshirt.dk/>
 *
 */

#include <security/pam_appl.h>
#include <security/pam_misc.h>
#include <stdio.h>
#include <unistd.h>
```

```
#include <gdbm.h>
#include <sys/types.h>
#include <sys/stat.h>

static struct pam_conv conv = {
  misc_conv,
  NULL
};


int main(int argc, char *argv[]) {

  pam_handle_t *pamh = NULL;  /** PAM data structure **/
  int retval;
  GDBM_FILE dbh;
  datum key, data;
  int flags;

  char *user = getlogin();

  /** Creating and initializing a PAM session **/
  retval = pam_start("vault", user, &conv, &pamh);
  if (retval == PAM_SUCCESS)
    /** Authenticate user **/
    retval = pam_authenticate(pamh, 0);

  if (retval == PAM_SUCCESS) {
    dbh = gdbm_open("vault.db", 512, GDBM_WRCREAT,
          S_IREAD|S_IWRITE, NULL);

    if (argc == 3) {
      key.dptr = strdup(argv[1]);
      key.dsize = strlen(argv[1])+1;
      data.dptr = strdup(argv[2]);
      data.dsize = strlen(argv[2])+1;
      gdbm_store(dbh, key, data, GDBM_REPLACE);
    } else {
        key.dptr = strdup(argv[1]);
        key.dsize = strlen(argv[1])+1;
        data = gdbm_fetch(dbh, key);
        printf("%s:%s\n", key.dptr, data.dptr);
      }
      gdbm_close(dbh);
    }

    fprintf(stderr, "%s\n", pam_strerror(pamh, retval));

    /** Destroy the PAM session **/
    pam_end(pamh, retval);
  }
```

The ssh_tunnels Module

The example of a small PAM module is the ssh_tunnels module. The module initiates a number of SSH tunnels for the user when he or she logs in. SSH tunnels tend to close when they are not used for a while, and the autossh program wraps the SSH client in order to prevent this (or more precisely—autossh will reconnect). The module use the autossh program instead of the plain SSH client.

```
/* pam_tunnels module */

/*
 * based on the pam_deny module (Linux PAM)
 */


#define PAM_SM_SESSION

#include <security/pam_modules.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
```

```
#include <pwd.h>
#include <syslog.h>

/* --- session management --- */

PAM_EXTERN int
pam_sm_open_session(pam_handle_t *pamh, int flags,
                    int argc, const char **argv)
{

  FILE *conffile;
  char *conffilename;
  const char *username = NULL;
  char *lport, *rport, *host;
  char cmdline[256], line[256];
  struct passwd *pwd = malloc(sizeof(struct passwd));

  if (pam_get_user(pamh, &username, NULL) != PAM_SUCCESS)
{
    syslog(LOG_ERR, "cannot determine user name");
    return PAM_USER_UNKNOWN;
  }
  pwd = getpwnam(username);
  conffilename = calloc(sizeof(char), strlen(
                 pwd->pw_dir)+20);
  sprintf(conffilename, "%s/.pam_tunnels.conf",
                 pwd->pw_dir);
  conffile = fopen(conffilename, "r");
  while ((fscanf(conffile, "%s\n", line)) != EOF) {
    lport = strtok(line, ":");
    host = strtok(NULL, ":");
    rport = strtok(NULL, ":");
    sprintf(cmdline, "autossh -f -N -L %s:%s:%s %s",
lport, host, rport, host);
    system(cmdline);
  }
  fclose(conffile);
  free(conffilename);
  return PAM_SUCCESS;
}
PAM_EXTERN int
pam_sm_close_session(pam_handle_t *pamh, int flags,
                    int argc, const char **argv)
{
  return PAM_SUCCESS;
}


/* end of module definition */

/* static module data */
#ifdef PAM_STATIC
struct pam_module _pam_deny_modstruct = {
    "pam_deny",
    NULL,
    NULL,
    NULL,
    pam_sm_open_session,
    pam_sm_close_session,
    NULL
};
#endif
```

Index

A

authentication problem 8

C

control flags
- about 26
- optional 28
- order 28
- required 27
- requisite 27
- sufficient 27

L

LDAP
- about 80
- client 81
- installing 81
- name service 82
- PAM, configuring 82

Linux PAM
- about 9
- compiling 10, 11
- downloading 10
- extra modules 12
- installing 9-12

M

management groups
- about 22
- account group 24
- auth group 23
- password group 25
- session group 25

modules, for restricted access
- about 58
- pam_access 60
- pam_deny 61
- pam_nologin 59
- pam_succeed_if 58
- pam_wheel 60

modules, related to background storage
- about 62
- pam_ldap 63
- pam_mysql 64, 65
- pam_unix 62
- pam_winbind 63

modules, related to user environment
- about 56
- pam_mkhomedir 56
- pam_mount 57
- uses, pam_mkhomedir 56

P

PAM
- about 7
- Apache htaccess 71
- authentication problem, solving 8
- authentication stages 22, 23
- configuration, consolidating 29, 30
- configuration, testing 37
- configuring with LDAP 82
- configuring with Winbind 78, 79
- control flags 26-29
- directory sevices 72
- encrypted home directories 67, 68
- environment, securing 32
- example 32-35
- extra modules, Linux PAM 12

  file system layout 15-19
  history 7
  implementations 12-14
  LDAP, directory services 80-82
  Linux box services 20, 21
  Linux distribution 13
  Linux PAM 9
  Linux PAM, compiling 10, 11
  Linux PAM, downloading 10
  management groups 22-25
  modules 56, 95, 103
  need for 9, 19
  PAM-aware applications 89
  parameters 51-55
  r-services, limiting 83, 84
  recipes 67
  Secure Shell 68
  stacking 26
  testing environment 37
  tests, automating 46, 47
  Winbind, directory sevices 73-79
PAM-aware applications
  about 89
  account health check 93
  conversation functions 94
  error messages 95
  example 90, 91
  PAM handling data structure, manipulating 93
  PAM session, closing 91, 92
  PAM session, opening 91, 92
  user, authenticating 92
  vault program 101-103
PAM configuration
  backdoor, leaving open 38
  bad example 47, 48
  enabling logging, get backstage 40
  get backstage 40-44
  log reading, get backstage 42-44
  pamtester utility 44-46
  test cases 39
PAM modules
  compiling 99
  management groups 96, 97
  related to background storage 62-64
  related to user environments 56, 57
  return codes 97
  Secure Shell tunnels 95
  ssh_tunnels modules 103-105
  supporting functions 98
  used to restrict access 58-61
PAM recipes
  Apache htaccess 71
  directory services 72
  encrypted home directories 67, 68
  r-services, limiting 83, 84
  Secure Shell 70
  Winbind, directory services 73-79
pamtester utility 44-46
parameters, PAM
  debug 52
  expose_account 55
  try_first_pass 55
  use_first_pass 53
Pluggable Authentication Modules. *See* **PAM**

S

Secure Shell
  about 68
  ssh-keygen utility 69
  working with 68
Secure Shell tunnels 95
SSH. *See* **Secure Shell**

W

Winbind
  about 73
  configuring 75, 76
  directory, joining 77
  Kerberos 76, 77
  overview 73-75
  PAM, configuring 78, 79